TWG-98-41 W. E. Burr
13 July 1998

Carlisle's Proposal on Multiple Algorithms and the Bridge CA

Carlisle Adams proposed at the July 13 TWG meeting that multiple signature certificates could be issued by the bridge CA to provide certification paths between a relying party who used one signature algorithm, and a signatory who used another. I impounded Carlisle's handwritten slides and said that I would write it up and post a description of the Proposal.

Appendix A restates all the applicable ASN.1, I believe. A certificate can be decomposed, or restated (in my pseudo ASN.1 notation) as:

```
Certificate      ::=      SEQUENCE {
        toBeSigned                        -- the body of the cert.
        SEQUENCE {
                algorithm                 -- simply an OID
                parameters OPTIONAL } --a sequence that depends on the algorithm
        ENCRYPTED-HASH { ToBeSigned }}  -- a bit string, that also depends on the
                                                          algorithm
```

In this case **toBeSigned** is the body of the certificate, but it doesn't really matter, it could be any message we want to sign, the technique isn't limited to certificates. Carlisle observes that the **algorithm** is just an OID, and we are free to introduce new OIDs into certificates, and several bodies have defined OIDs for different algorithms; indeed there is an overabundance of such OIDs. So we are not doing any violence to X.509, if we add another, call it **multipleAlgorithms**.

The interesting thing is that **parameters** and **ENCRYPTED-HASH** all entirely depend on the **algorithm OID**. Carlisle exploits this by proposing to define another algorithm OID, **multipleAlgorithms**. The associated **parameters** is a list of AlgorithmIdentifier (i.e., a list of {OID, Parameter} pairs), of several individual algorithms used to sign the certificate. And **ENCRYPTED-HASH,** which is a bit string, now becomes a sequence of bit strings (the whole sequence then encoded as a BIT STRING, to comply with traditional syntax), each one corresponding to a different signature on the certificate, in the same order as the AlgIds given in **parameters**.

This would create sort of super certificate to be issued by the BCA to PCA,s certifying the PCA's key with all the algorithms the BCA supports. Only one multiple certificate is needed per PCA.

If we have taken over parameters for this purpose, where do we get the parameters used to validate the signatures for DSA or ECDSA? This bothered me at first. It turns out that we should never get parameters from the field of a signature; that is insecure because the signature does not protect the parameters (and it would be circular if it did). We get the parameters we need to validate the signature from the **subjectPublicKeyInfo** of certificate the relying party's PCA issued to the BCA, just as we would for any other signature. It is important to understand that only one of the signatures, that for the algorithm used by the relying party's PCA, would be validated by relying party, when the multiple algorithm certificate is used.

Carlisle has not actually changed the ASN.1 definition of a certificate at all. He has simply added an new (somewhat unusual) signature algorithm, which is a list of other algorithms, but which everybody needs to process (at least for validation) if this is to work. Carlisle's approach has some advantages:

- It eliminates the complexity in finding involved in finding the “correct” one of several mixed algorithm certificates issued by the BCA to each PCA in my “multiple algorithm BCA” approach;
- It eliminates the complexity inherent in finding the “correct” one of several mixed algorithm certificates issued by each PCA to the PCA in Santosh’s variation;
- As compared to the “split bridge” approach, it eliminates the complexity of splitting the Bridge into several CAs, and eliminates one additional certificate in each path.
- It eliminates the complexity of keeping multiple mixed algorithm certificates in synch (with respect to things like expiration) for any of the alternatives above;

The singular disadvantage of Carlisle’s proposal is that it requires the adoption of a new, fairly unusual algorithm identifier by most of the client vendors, before it will work. Carlisle argues (correctly, I believe) that the actual complexity of this is much less than adding the ability to validate a second or third signature algorithm, which is required in any case. But it is extra, and perhaps a big assumption.

One factor that I need to think some more about is the general utility of this multiple signature technique. We often have multiple signatures on paper documents. Surely there are other applications for a multiply signed document. My intuition is that it may be hard to justify introducing this just for multiple algorithms and the bridge CA, which has other, if probably less desirable solutions, but it may be worthwhile if we can identify other important applications of the technique.

Appendix A - The ASN.1

```
Certificate                ::=    SIGNED { SEQUENCE {
      version                     [0]    Version DEFAULT v1,
      serialNumber                CertificateSerialNumber,
      signature                   AlgorithmIdentifier,
      issuer                      Name,
      validity                    Validity,
      subject                     Name,
      subjectPublicKeyInfo        SubjectPublicKeyInfo,
      issuerUniqueIdentifier      [1]    IMPLICIT UniqueIdentifier OPTIONAL,
                                                 -- if present, version must be v2 or v3
      subjectUniqueIdentifier     [2]    IMPLICIT UniqueIdentifier OPTIONAL
                                                 -- if present, version must be v2 or v3
      extensions                  [3]    Extensions OPTIONAL
                                                 -- If present, version must be v3 -- } }
```

```
AlgorithmIdentifier               ::=    SEQUENCE {
      algorithm          ALGORITHM.&id ({SupportedAlgorithms}),
      parameters         ALGORITHM.&Type ({SupportedAlgorithms}{ @algorithm}) OPTIONAL }
```

--   *Definition of the following information object set is deferred, perhaps to standardized*
--   *profiles or to protocol implementation conformance statements. The set is required to*
--   *specify a table constraint on the* **parameters** *component of* **AlgorithmIdentifier**.

```
SupportedAlgorithms     ALGORITHM   ::=      { ... }
```

```
ENCRYPTED-HASH { ToBeSigned }     ::=      BIT STRING ( CONSTRAINED BY {
      -- must be the result of applying a hashing procedure to the DER-encoded (see 8.7) octets --
      -- of a value of -- ToBeSigned -- and then applying an encipherment procedure to those octets -- })

ENCRYPTED { ToBeEnciphered }      ::=      BIT STRING ( CONSTRAINED BY {
      -- must be the result of applying an encipherment procedure --
      -- to the BER-encoded octets of a value of -- ToBeEnciphered})
```

```
SIGNATURE { ToBeSigned }          ::=      SEQUENCE {
      algorithmIdentifier         AlgorithmIdentifier,
      encrypted                   ENCRYPTED-HASH { ToBeSigned }}
```

```
SIGNED { ToBeSigned }             ::=    SEQUENCE {
      toBeSigned                  ToBeSigned,
      COMPONENTS OF               SIGNATURE { ToBeSigned }}
```